ECO-AI Hackathon Track 2

Deep Learning Emulators of Coupled Time-Dependent PDEs

Carbon Hackers (ChatGPT-recommended)

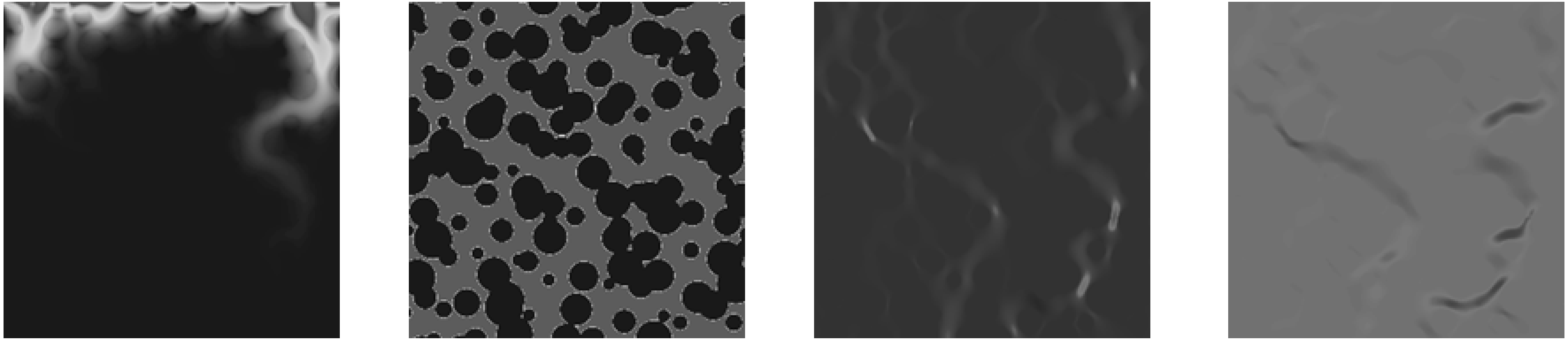
Jack Li
Rui Li
Vitalii Starikov
Donghu Guo
Farah Rabie

Introduction

Objective Develop a machine-learning based emulator for reactive transport simulations

Can a machine-learning based emulator accurately predict the following fields?

concentration porosity Ux Uy



Data

16 GeoChemFoam simulations, each initialized with a different random distribution of porosity

12 datasets for training and 4 for validation

Baseline Model

U-net Residual Network (ResNet) 34

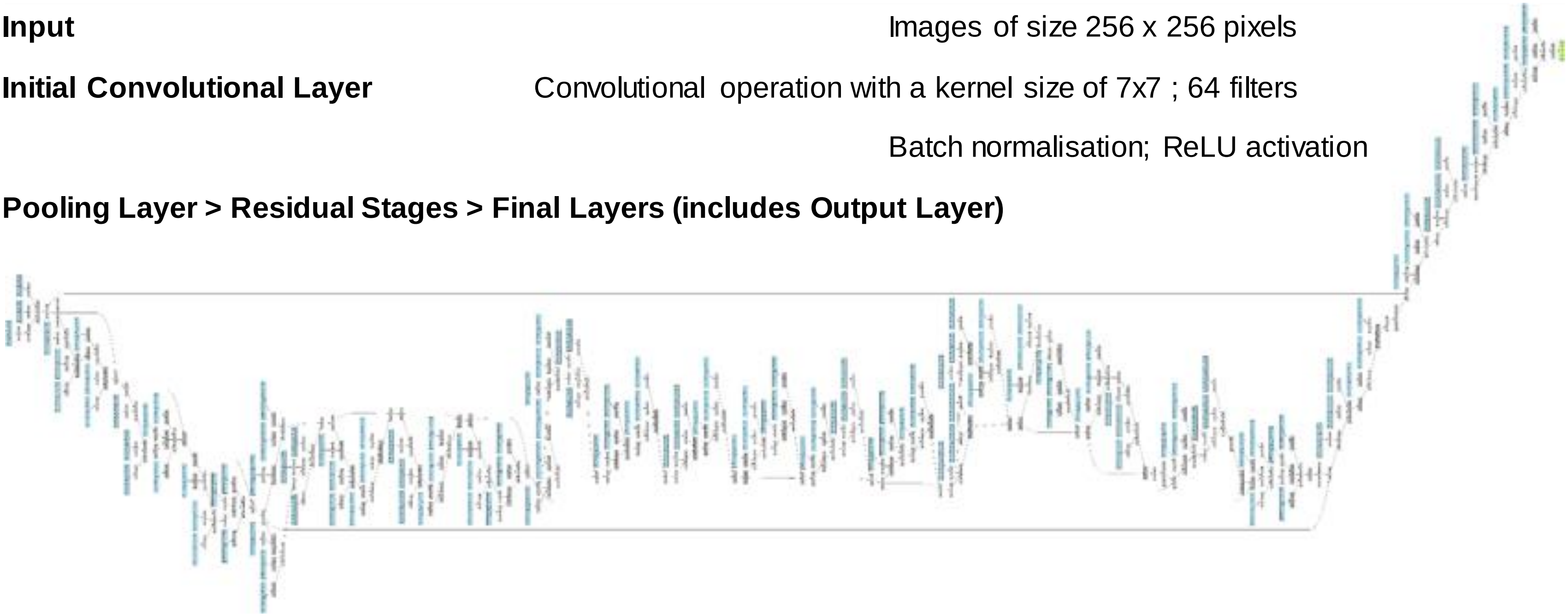
Network Type Convolutional Neural Network

Input Images of size 256 x 256 pixels

Initial Convolutional Layer Convolutional operation with a kernel size of 7x7 ; 64 filters

Batch normalisation; ReLU activation

Pooling Layer > Residual Stages > Final Layers (includes Output Layer)



Baseline Model

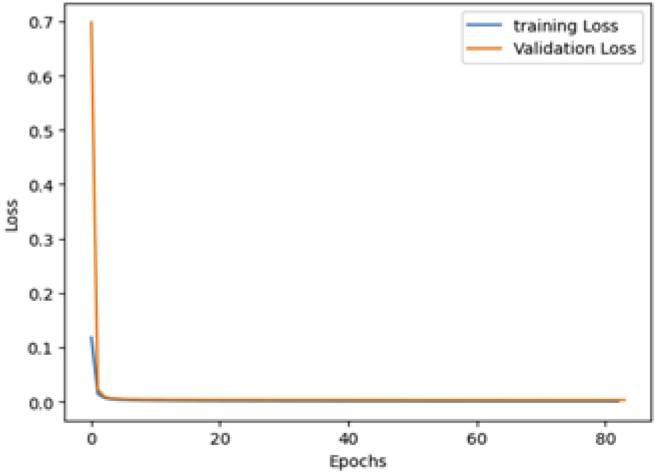
U-net Residual Network (ResNet) 34

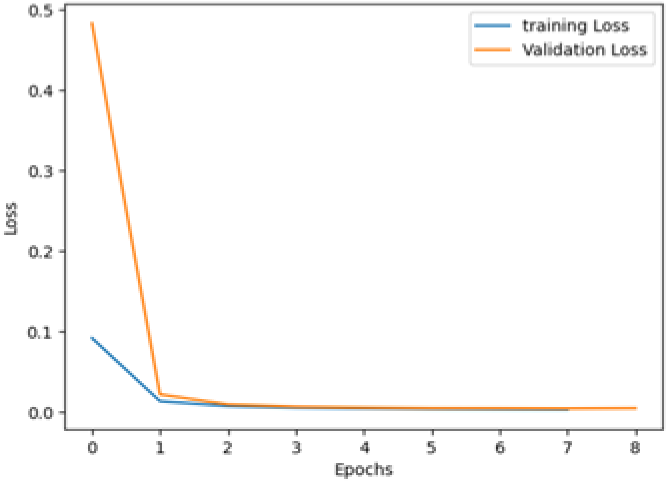
Input 10 simulations (2 simulations for validation)

No. of batches 64

Output

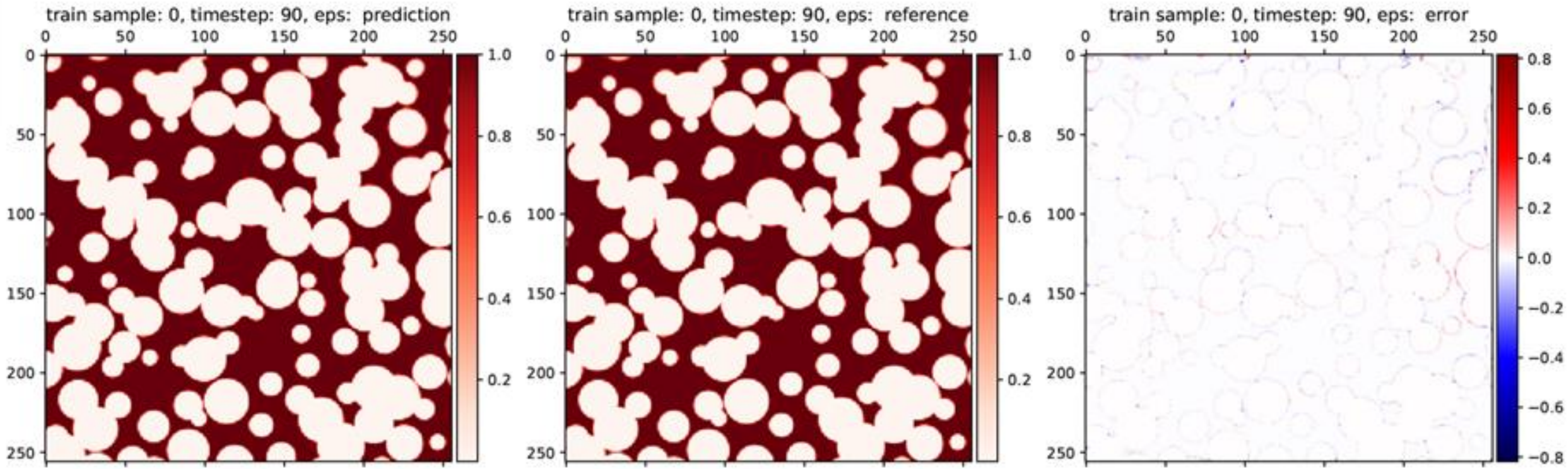
Test metric	DataLoader 0
test_loss	0.004011622630059719



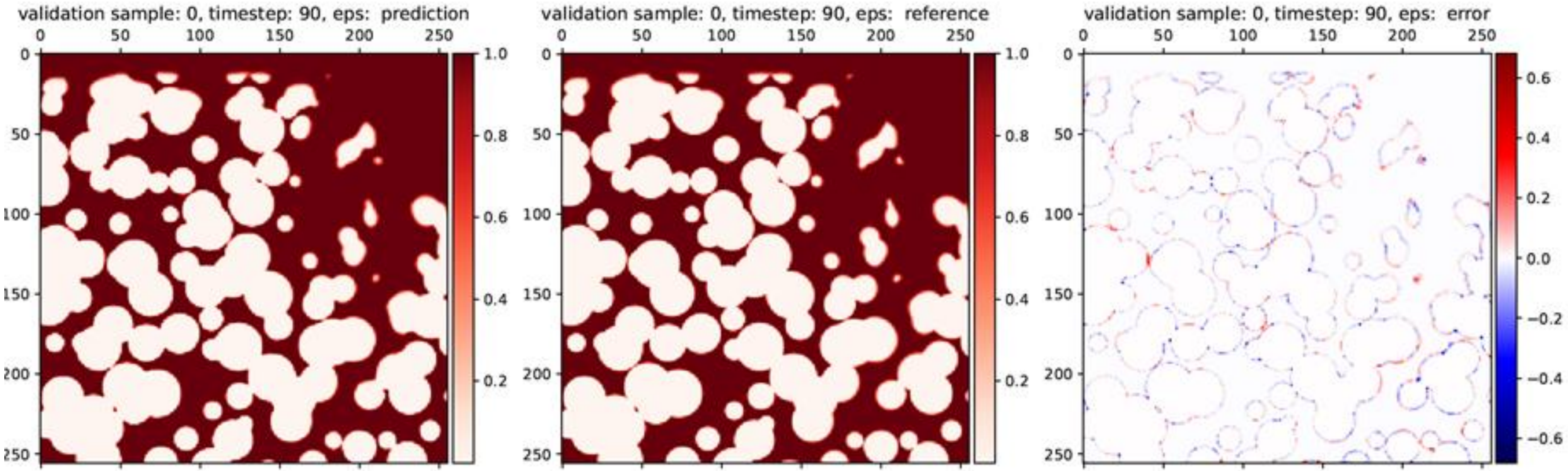


157/157 [01:42<00:00, 1.53it/s, v_num=1, train_loss_step=0.00244, val_loss=0.00351, train_loss_epoch=0.00242]

Baseline Results



Baseline Results



Alternatives

CNN Model

Architecture of Neural Network

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 16, 256, 256]           1,168
         LeakyReLU-2        [-1, 16, 256, 256]               0
            Conv2d-3        [-1, 48, 256, 256]           6,960
         LeakyReLU-4        [-1, 48, 256, 256]               0
            Conv2d-5       [-1, 128, 256, 256]          55,424
         LeakyReLU-6       [-1, 128, 256, 256]               0
   ConvTranspose2d-7        [-1, 48, 256, 256]          55,344
         LeakyReLU-8        [-1, 48, 256, 256]               0
   ConvTranspose2d-9        [-1, 16, 256, 256]           6,928
        LeakyReLU-10        [-1, 16, 256, 256]               0
  ConvTranspose2d-11         [-1, 4, 256, 256]             580
        LeakyReLU-12         [-1, 4, 256, 256]               0
         Identity-13         [-1, 4, 256, 256]               0
================================================================
Total params: 126,404
Trainable params: 126,404
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 2.00
Forward/backward pass size (MB): 262.00
Params size (MB): 0.48
Estimated Total Size (MB): 264.48
----------------------------------------------------------------
```

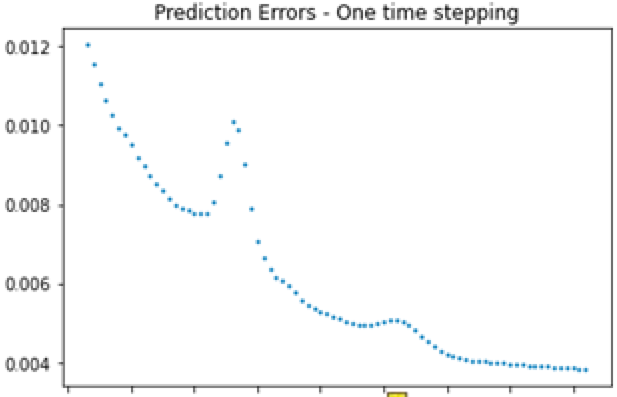
```
Sample size
(1, 8, 256, 256)
(1, 4, 256, 256)

~80 samples for one simulation

The first simulation for training
The 15th simulation for testing
```

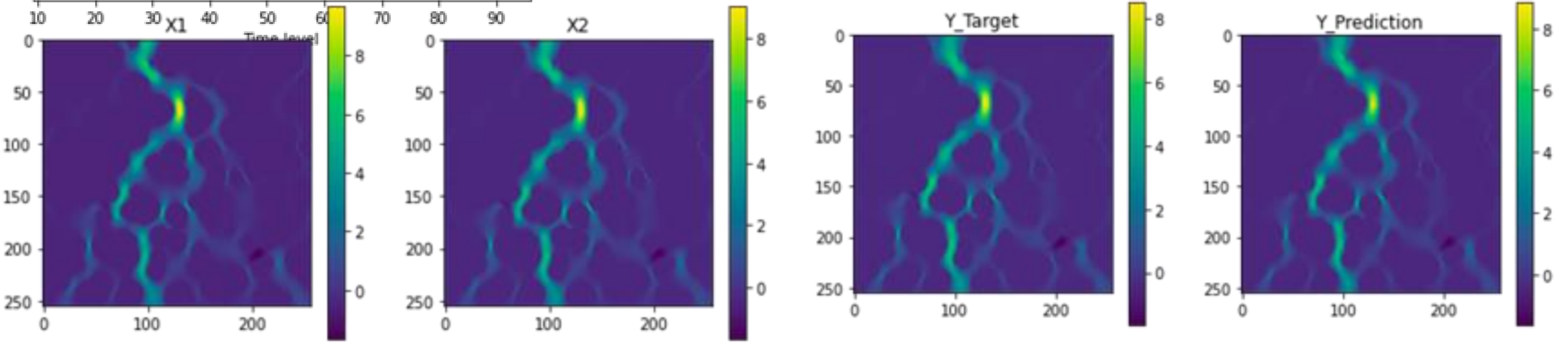
Alternatives CNN Model

A/ One-time stepping



prediction error: 0.008706905418288828 at
timelevel: 34

Velocity in x-direction



Alternatives

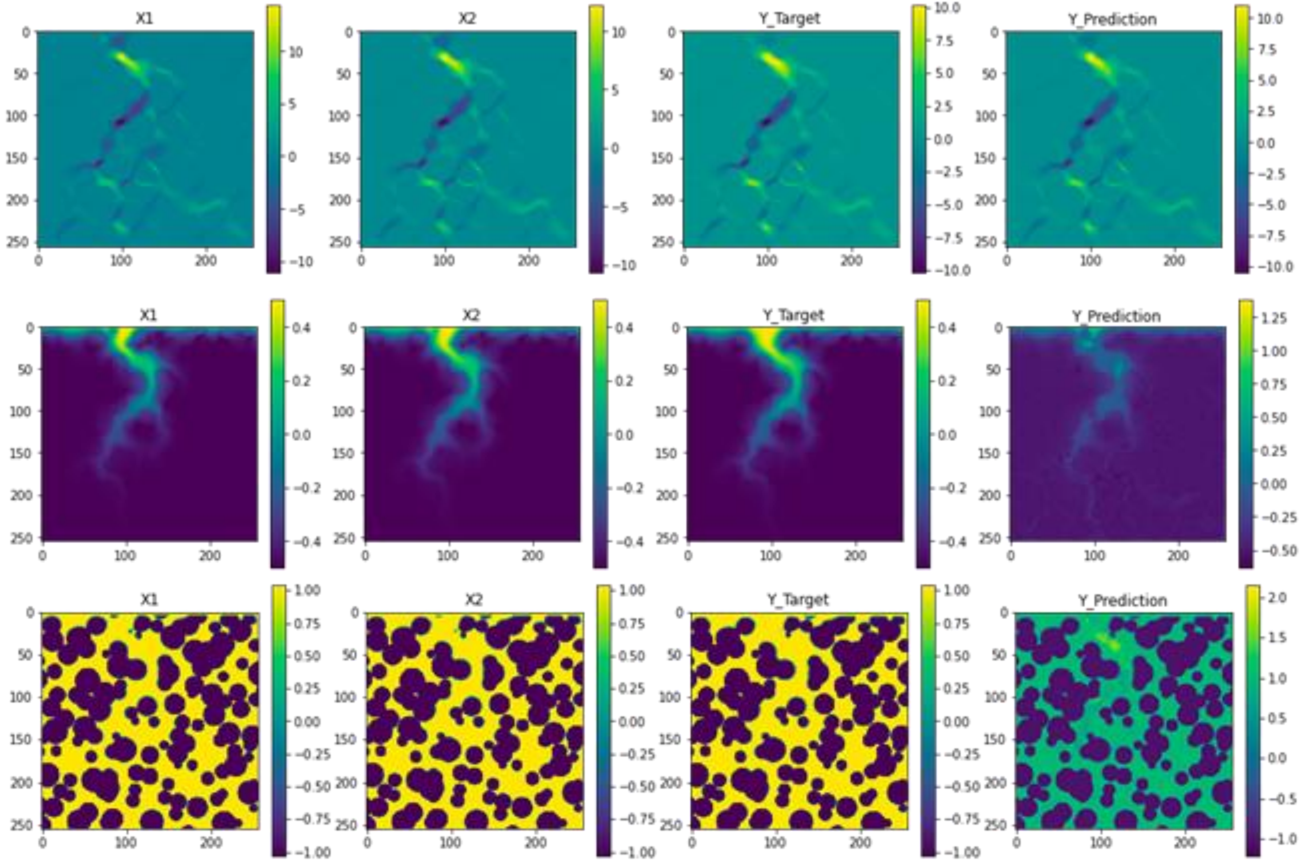
CNN Model

A/ One-time stepping

Velocity in y-direction

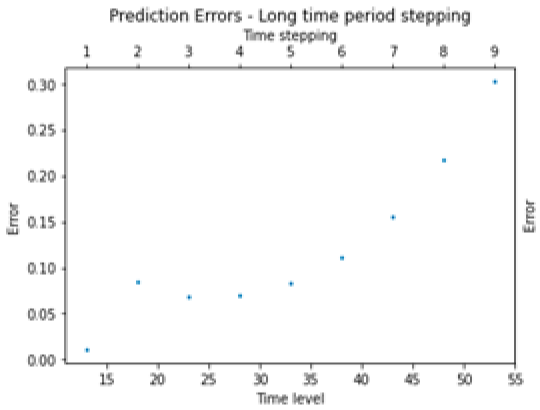
Concentration

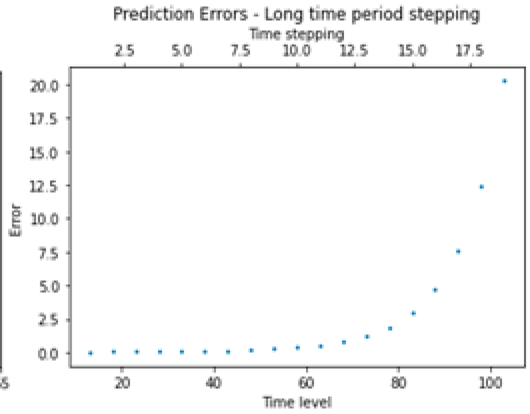
Porosity



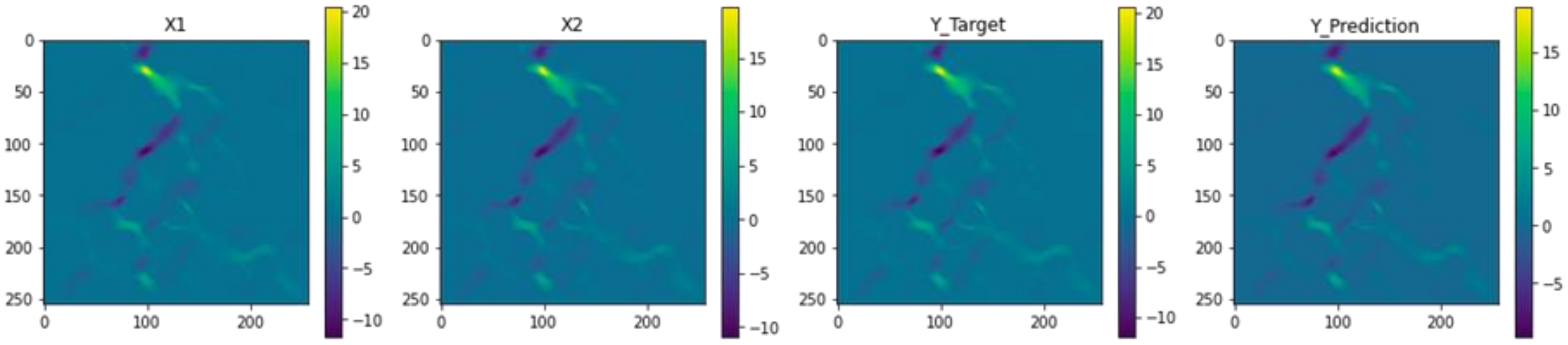
Alternatives CNN Model

B/ Multi-time stepping; autoregression






```
prediction error: 0.06933388550747517
time level: 28 time stepping: 4
```



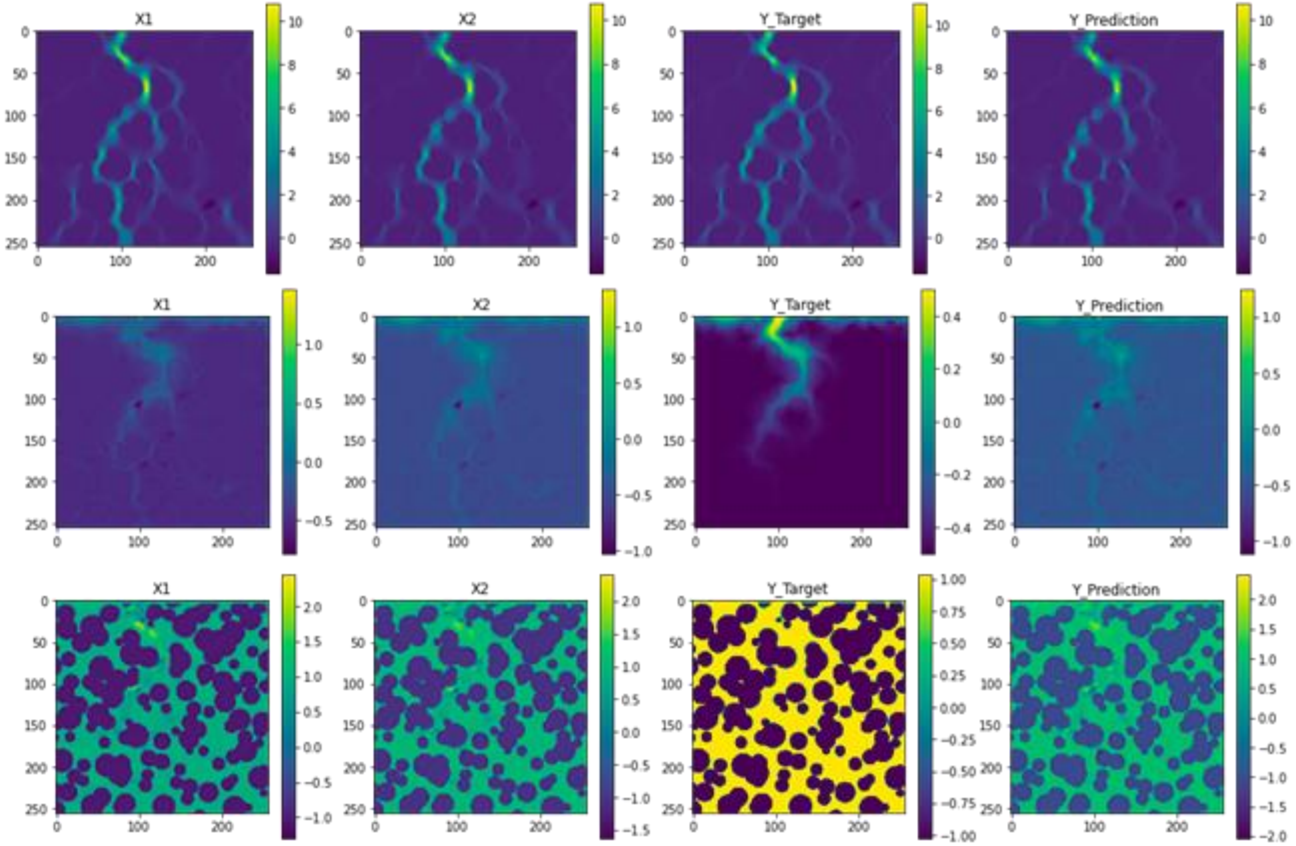
Alternatives CNN Model

B/ Multi-time stepping; autoregression

Velocity in y-direction

Concentration

Porosity

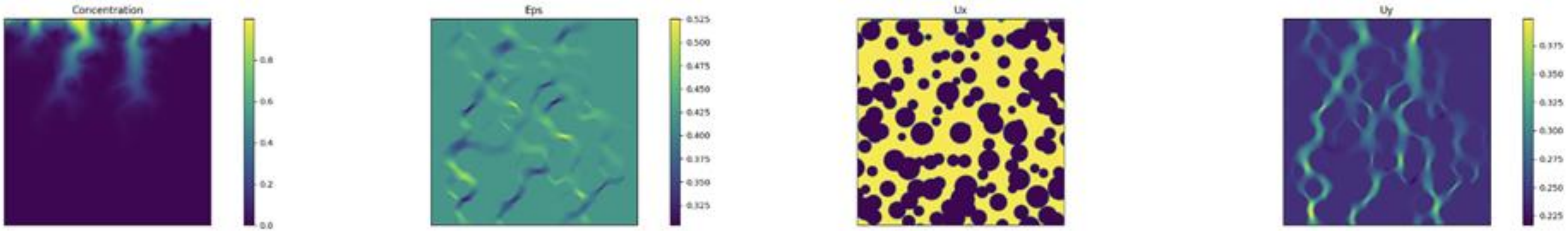


Alternatives ConvLSTM (several times of trials)

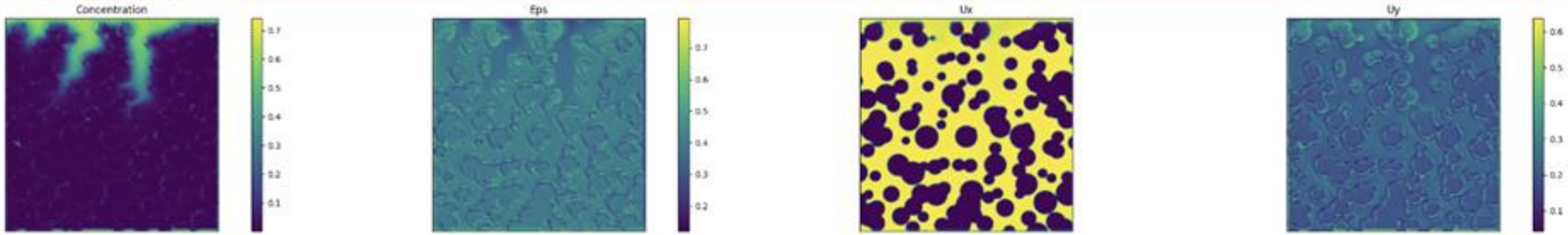
Using N previous time steps to predict future one step

Treating C, eps, Ux, and Uy as separate channels

Previous 10 steps — the 10th step

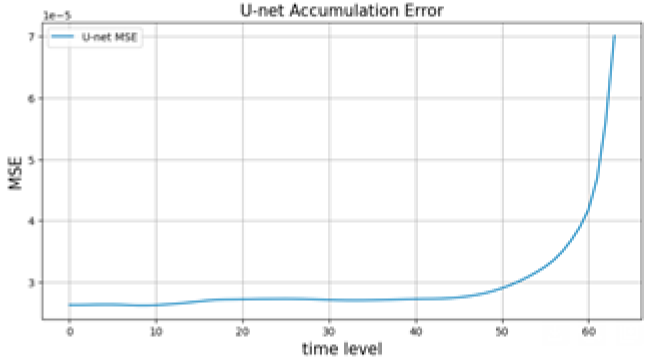


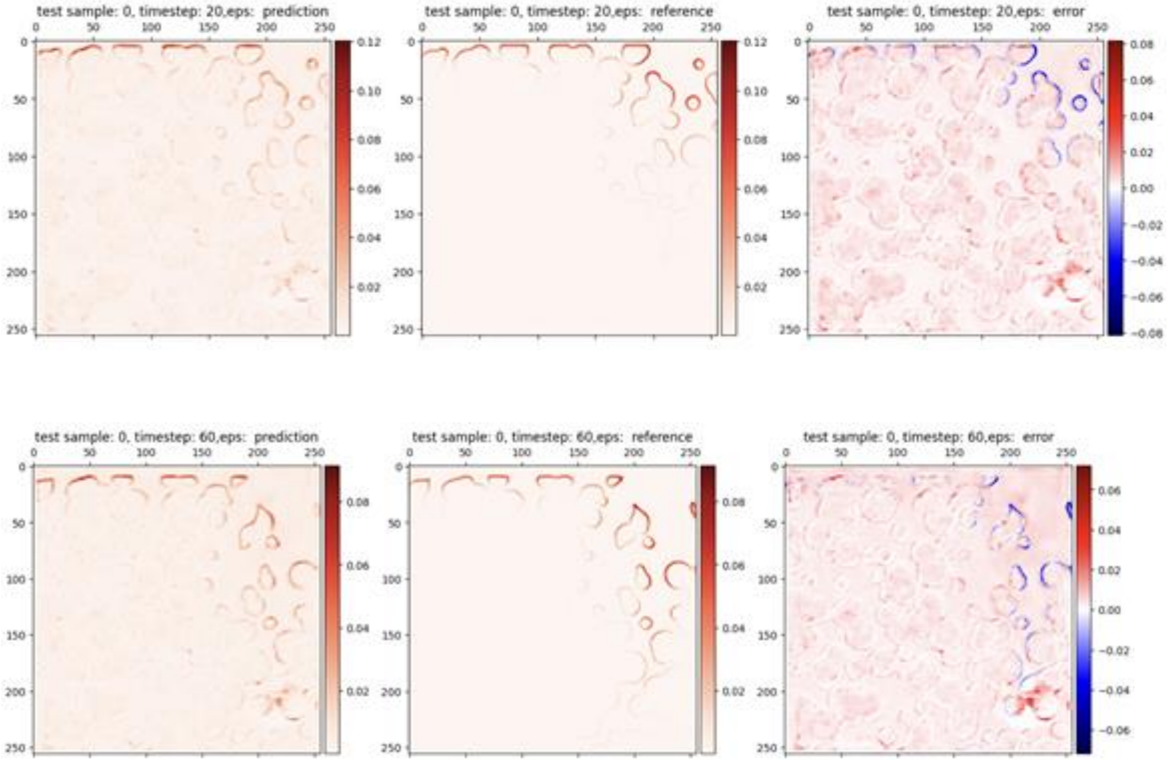
Predicted 11th step



Alternatives Auto-regressive U-Net

Takes all 4 fields as input at a previous time step, and predicts their change for the next time step





Conclusion

- This is a difficult task – working with large datasets can be quite challenging
- Regardless of learning algorithm, validation error hardly decreases during training

Thoughts

- How can we pre-process the data, to help deal with memory issues?
- Which variables should we be trying to learn (are they all varying significantly)?
- How do we decide on which timesteps to include (should we include gaps in timesteps)?
- Can we use Physics-based Machine Learning algorithms for this task?